

[0038] OPLs are a data structure for saving data in a data file. More detailed descriptions of OPLs can be found in U.S. Pat. Nos. 5,933,842 and 5,946,696. The disclosure of those patents is hereby incorporated by reference.

[0039] Conventional OPLs provide a data structure for a data file. That data structure enables the storing of information in a common format useful to provide compatibility with previous and future versions of an application program. The format of the OPL enables skipping over data that the application program does not recognize while using the data that the application program does recognize. Conventional OPLs can grow with new information in a new version of an application program because they can be dynamically expanded for new objects and properties.

[0040] Referring to FIG. 3, characteristics of a conventional OPL 300 will be described. (For additional information regarding OPLs, see U.S. Pat. Nos. 5,933,842 and 5,946,696.)

[0041] An "object," as used in this disclosure, refers to an entity that has characteristics and that is either displayed by an application program or is part of an application program. An example of an object displayed by an application program is a text box contained in a window of an application program into which a user can input text. The characteristics of the text box can include its color, the font of the text, and the point size of the text. An example of an object that is part of an application program is an in-memory representation of an animal, where its characteristics can include its color, number of legs, and whether it is a carnivore. This in-memory representation can be implemented as a data structure with the elements of the data structure storing the characteristics. An example of such a data structure is the C++ class data structure. The characteristics of an object are referred to as properties of the object. Each property of an object typically has a value. For example, the color property may have the value red.

[0042] As discussed in the Background section, an example of an object 100 is illustrated in FIG. 1. Object 100 can comprise a border 102 and text 104. Object 100 can have six properties for border 102 and text 104. Border 102 can have properties of border style and border size. Text 104 can have properties of font, text size, justification, and text style. The default values for border 102 and text 104 can be that the border style is solid, the border size is 4 point, the text font is Times New Roman, the text size is 20 point, the text justification is left, and the text style is non-italic.

[0043] Since OPL 300 uses a standard format, it can be used to internally store the properties of an object. For example, OPL 300 can include an object property identification ("opid") element 302, an object property type element 304, and an object property value element 306. Opid element 302 can contain a numerical identifier that is associated with a particular property. The system can maintain a mapping of all properties to their opid, and this mapping can vary from OPL to OPL. In other words, each OPL can define what property the opid values represent and other OPLs can use the same opid values for different properties. Object property type element 304 refers to a data type, such as Short Integer (2 bytes), Long Integer (4 bytes), Unicode string, etc. Object property value element 306 conforms to the appropriate data type of the associated object property type element 304. As shown in FIG. 3 for

example, opid "1" denotes an object property type element 304 of Short Integer (2 bytes), which the system knows corresponds to a "color," and an object property value element 306 of 0x00FF0000, which is the RGB (Red, Green, Blue) encoding for "red." When referring to a particular property, the opid is used. For example, if the border property in OPL 300 is desired, it is referenced by its opid "2." An OPL may also contain another OPL as a property, allowing for more complex data structures to be created.

[0044] In a 16-bit conventional OPL, opid element 302 is typically eleven bits. In such a configuration, OPL 300 is limited to 2048 items because of the eleven bit constraint. In a publishing system, the number of objects and properties can easily exceed 2048 items. A publishing document can have multiple pages, each having many objects. Each object can have many properties associated with it. Accordingly, maximizing OPL capacity in some instances is desirable. To this end, the present invention can include an improved OPL type, the OPL array, having an almost unlimited capacity. OPL arrays can store large amounts of data, such as all objects in a document. An OPL can still be used to store smaller quantities of properties.

[0045] Referring to FIG. 4, a Data structure 400 according to an exemplary embodiment of the present invention will be described. Data structure 400 can include root OPL 402. Root OPL 402 can include a "Max" property 402a, an invariant property 402b, and an OPL array property 402c. While not necessary, max property 402a can indicate the size of OPL array property 402c and can provide the convenience of allocating adequate memory when reading data structure 400. Invariant property 402b can define a feature where the array indices of items in OPL array property 402c remain constant. In other words, the array index of an item in OPL array property 402c will not change throughout the lifetime of the item. Accordingly, new items can only be added at an index that is not currently used, and deleting an item results in an empty index location.

[0046] OPL array property 402c can reference OPL array 404. OPL array 404 can be stored in OPL property 402c, or it can be stored separately. OPL array 404 can contain properties, other OPLs, or other OPL Arrays. As shown in FIG. 4, OPL array 404 can include a sub-object property list (a "subopl") 404a-404e, where each subopl 404a-404e can be a property, an OPL, or another OPL Array. Each subopl 404a-404e can reference an array element 406. Each array element 406 can be any property, an OPL, or another OPL Array. In the example, array elements 406 are OPLs that contain an ID and a variable length string.

[0047] While only one OPL array 404 is shown in FIG. 4, with its associated OPL array elements 406, the present invention is not limited to such a structure. For example, data structure 400 can include a plurality of OPLs like OPL array 404, each having associated OPL array elements 406. In that case, index 402c can reference each OPL array 404.

[0048] Subopls 404a-404e each reference a property, object, or other OPL similarly to opid 302 (FIG. 3) of a conventional OPL. However, in data structure 400, the items in OPL array 404 are not given a specific opid. Instead, the opid for each subopl 404a-404e in OPL array 404 can be set to "-1." Then, the position of each subopl 404a-404e in OPL array 404 can be used as the opid. The positions of each subopl 404a-404e are represented in FIG. 4 by respec-